1

Taking the Metro with Windows Phone

WHAT'S IN THIS CHAPTER

- How Windows Phone has changed Microsoft's approach to the mobile industry
- What the Metro Design Language is and how it came about
- An overview of the Start and Lock Screens and how they help users access information on the phone
- Why the use of Hubs creates a more connected user experience
- What it means to be a Windows Phone developer

Microsoft has been building mobile devices for well over 10 years, starting with a variety of Windows CE-based devices, such as the Handheld PC and the Palm-size PC, first released in 1996. Beginning around 2000, these disparate operating systems began converging into what became Windows Mobile, based on the principle of delivering a PC to your pocket. New features were predominately driven by enterprise needs such as device management and security. This eventually worked to the detriment of the platform as it didn't appeal to the average consumer. Devices were more robust than sexy, and the user interface mirrored that of the desktop, even having a Start menu, rather than providing an experience.

Throughout this chapter, and in other parts of this book, there will be references to both Windows Mobile *and* Windows Phone. *This is intentional, and they are not the same thing.* Windows Mobile *refers to the previous mobile operating system from Microsoft that at the time of writing is Windows Mobile 6.5.3.* Windows Phone *refers to Microsoft's latest offering in the mobile space and starts with Windows Phone 7.*

In February 2010 at the Mobile World Congress, Microsoft unveiled the Windows Phone 7 series, a new-look mobile operating system featuring hard edges, full bleed pages, and sharp typography. Code-named *Metro*, the new look-and-feel is more akin to an immersive experience than to a collection of mobile applications. It appears that Microsoft has taken several leaves out of the playbooks of other mobile platforms, while still innovating and delivering on their own set of values and practices.

Before launching into how you can get started building applications for Windows Phone, it is important to understand the Metro user experience. This will help you build applications that not only run on a Windows Phone but also integrate into the mobile experience.

MINIMUM SPECIFICATIONS

Traditionally, Windows Mobile has had the stigma attached to it that it is slow and unreliable. This had little to do with the underlying operating system, but rather the other stakeholders involved in getting a device into market. Manufacturers, telecommunication companies, application developers, and other third parties all contribute to what comes prepackaged on a device. Each one of these parties builds or adds features that they think will benefit the user. Unfortunately, quite often these features either degrade the overall experience — for example, hogging precious device resources — or don't play well with other aspects of the phone. This has led to an overall negative impression of the Windows Mobile platform as a whole.

Microsoft took the opportunity with Windows Phone to restructure the ecosystem in which devices operate. Although they haven't been so arrogant as to come out with the *Microsoft Phone*, which many were anticipating, they have put some checks and balances in place to ensure that users receive an amazing experience, and, furthermore, that this experience is uniform throughout the phone and for the duration of the phone's life.

It all starts with the hardware. Previously Microsoft has been overly optimistic in specifying the minimum specifications for Windows Mobile. This resulted in many devices that were woefully underpowered, and although this kept the price point low, the devices were frustratingly slow and unresponsive to use. Going forward, Microsoft has defined a much higher set of minimum hardware requirements for Windows Phone, which includes a 1-GHz processor and support for graphics hardware acceleration. When you look at the frameworks that are to be used to develop applications and games for this platform, it is very evident why such high hardware specifications are required.

In addition to having graphics acceleration, Windows Phone devices will appear to be highly responsive because of the use of capacitance screens. This, in turn, lends itself to supporting multi-touch. The net effect is that users will interact with a Windows Phone device using gestures such as tap, pinch, and swipe with their fingers, rather than the more traditional mechanism of using a stylus.

There is currently no intention to support a non-touch-screen Windows Phone device. However, device manufacturers will still be able to differentiate their devices through different device ergonomics and the optional inclusion of a hardware keyboard. A hardware keyboard will complement the Windows Phone experience, making it easier to enter text rapidly. This is particularly useful for e-mail, messaging, and annotating documents on the road.

Chassis Design

So far you know that hardware manufacturers can optionally include a keyboard, but what else can they modify? In the past this was quite an open-ended discussion as manufacturers could build a device to meet a certain price point. For example, for high-end devices, they could include GPS, an accelerometer, and a high-resolution camera; a low-end device may only have a T9 keypad and no camera at all. Even the number and layout of physical hardware buttons could change between devices. Of course, all these options come at a cost, and the first place this took hold was with developers. When building applications, developers were seldom able to rely on a particular hardware feature being present. Instead, you would typically either query an API to determine if hardware existed, or simply attempt to address the hardware. Failure or an exception would indicate the lack of supported hardware.

After the application had been developed, the problem was transposed to the end users. They would see a product advertised as being compatible with Windows Mobile and purchase it, only to discover that it required hardware that they didn't have. No two Windows Mobile devices were alike. When it came to Marketplace for Windows Mobile, Microsoft acknowledged this issue, and as part of application submission, developers had to indicate what device capabilities their application required. The Marketplace client running on the device would then restrict the list of applications to only those that matched the device capabilities.

For Windows Phone, Microsoft has taken the proactive position of enforcing a set of requirements around device capabilities. This has been achieved by taking the traditional minimum hardware specifications and turning them into what Microsoft calls a *chassis design*. This specifies the external buttons, and in some cases their location, and the inclusion of particular hardware features such as Wi-Fi, GPS, accelerometer, compass, camera, light and proximity sensors, and the ability to vibrate. A device that doesn't include all of the features dictated by a chassis design cannot be called a Windows Phone.

On the front-facing side of a Windows Phone there will be three buttons: Back, Start, and Search. There will also be dedicated camera, power, and volume controls. Figure 1-1 shows an example device in both portrait and landscape, illustrating the relative positions of the three front-facing hardware buttons.

FIGURE 1-1

It's important to note that these hardware buttons have a dedicated purpose. Unlike in Windows Mobile, where the buttons could be assigned by the user to different functions, and then applications could elect to override all or some of the buttons, on a Windows Phone the buttons have a sole purpose. This reinforces the overall user experience through a consistent interface.

The one exception to this rule is the Back button. Within your application you are able to control the navigation sequence. This means that you are able to intercept and handle the Back button. However, it is important to remember that the purpose of this button is to navigate back to whence the user came. For example, if they click to delete an item from a list and the application displays a confirmation prompt, the Back button should dismiss this prompt without deleting the item. Similarly, if the user has clicked an item in a list and gone through to a Details view, the Back button should navigate the user back to the list of items.

If you ensure that you correctly handle the Back button, there should be little need for your application to include navigation controls within the context of your application. Forward navigation is typically done through interaction with content; Back navigation is instigated from the Back button, and exiting your application is little more than pressing the Back button when on the first page of the application.

You can think of every application you open as being placed on a stack. When you hit the Back button and have not purposely handled it within your application, the application is popped off the stack and the previous one is displayed. This analogy works well as Windows Phone will automatically close your application when it goes out of focus by being popped off the application stack.

Similar to other mobile platforms, Windows Phone has a dedicated "I'm lost, take me to a known location" button. As it's a Microsoft platform, this button is logically called the *Start button* and takes the user back to the Start experience. As you will learn, the Start is an area on the device that contains a personalized set of tiles that reflect what's important to the user. It also acts as a launching point for accessing areas of the device and applications that the user may have installed.

The last of the buttons on the front of the device is the Search button, also known as the *Bing button*. Pressing the Search button launches a context-sensitive search. For example, if you are looking through your contacts, pressing the Search button will filter your contacts based on the search criteria. If there is no appropriate search context, pressing the Search button will launch Bing Search, allowing you to search over Web content, images, and maps. At this stage it is not possible to integrate the Search button into your application, so tapping it within any third-party application will launch Bing Search.

The inclusion of Wi-Fi seems like an obvious requirement, but with the advent of 3G+ networks that are continuing to get cheaper, it would have been an easy cost saving for manufacturers to omit the Wi-Fi stack. In the early days of Windows Mobile, before Microsoft tightened security, it used to be possible to synchronize your contracts, calendar, and e-mail with Outlook by connecting to ActiveSync through a Wi-Fi network. This capability is returning with the ability to synchronize across your home Wi-Fi network to your Zune desktop experience.

Location is definitely one of the hip new fads being talked about across the software development community. Software that is aware of the user's location means that it can locate information and people nearby. Of course, there are all manner of privacy issues to navigate, but it is important that

Windows Phone can provide location information. This topic will be covered in detail in the context of the location services offered by the platform in Chapter 12, but it's enough to say that having a GPS is essential in order to accurately geolocate the user.

One thing that you will notice about Windows Phone is that it is the first mobile offering from Microsoft that has been designed with a consumer rather than enterprise or business user focus. Previously, Windows Mobile was more tailored for the mobile worker, with support for enterprise features such as device deployment and management at the expense of a consistent set of hardware capabilities. As a consumer device Windows Phone will offer a minimum of a 5-megapixel camera with integrated flash. Windows Phones will also include light and proximity sensors that will be used to enhance the user experience.

In building your application, you need to be very aware of the experience you are constructing for the user. Where you would have once provided simple on-screen feedback, you can now use more complex animation and sounds and even have the device vibrate. You should use all visual and hardware effects sparingly as it is easy to overwhelm the user and drain the phone's battery in the process.

Screen Resolution

Dealing with differing hardware was just one of the challenges faced by developers working with the Windows Mobile platform. In seeking to be a platform that could be tailored to a wide variety of user scenarios, Windows Mobile 6.5 supported six touch-screen and five non-touch-screen resolutions. As you have already learned, there won't be support for non-touch screen in Windows Phone, but what's even more exciting for developers is that with Windows Phone there will only be two different screen resolutions that you need to accommodate.

The initial platform will be released with WVGA (480 × 800) resolution. A second resolution, HVGA (320 × 480), will follow sometime in the future.

Whenever you see a Windows Phone being demonstrated, it is likely that it will be in Portrait mode. In fact, if you look at some of the core areas of Windows Phone, such as Start, they only support being displayed in Portrait mode. However, this doesn't prevent you from taking advantage of running in Landscape if that's more suitable for your application. In fact, the best applications are those that allow either orientation, reorganizing the layout in order to make best use of the available screen real estate. Windows Phone also provides the necessary extension points for your application to handle the change in device orientation during operation, such as sliding out a physical keyboard on a device that has one.

METRO DESIGN LANGUAGE

Before getting much further into the ins and outs of Windows Phone, it's worth taking a step backward and looking at the approach that Microsoft took in designing the user experience. Rather than simply refining what they already had, it was time to make a clean break and come up with a revolutionary design. The outcome of this process was not only a unique mobile interface but also a design language that you can, and should, adopt as part of building your application.

As part of going forward, it's important to look back, even if only for a moment to wave as it disappears into the sunset. In the case of Windows Phone, you can see from Figure 1-2 the legacy of Windows Mobile from Pocket PC 2003 SE, through Windows Mobile 5.0, 6.1.4, and finally 6.5.3. As you can see, each has been an incremental approach with little to excite the user, save the more touch-friendly home screen and controls in Windows Mobile 6.5.3.

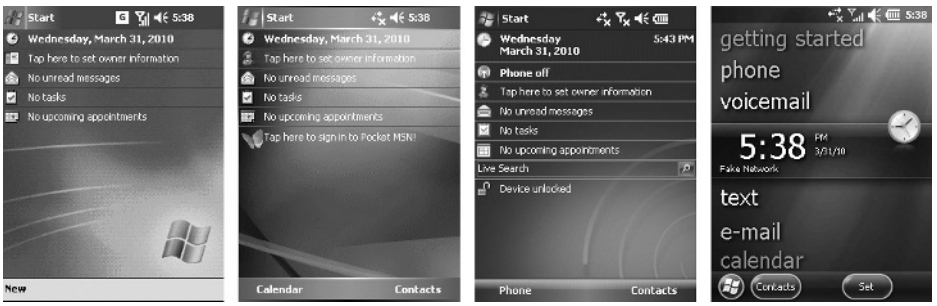

FIGURE 1-2

What's important to recognize is that although the Windows Mobile user interface appears somewhat dated now, there are some important concepts embraced within the layouts — for example how relevant information could easily be accessed right from the home screen, and that applications were only a click or two away via the Start menu.

The user interface on nearly all the current-generation smartphones is geared around applications. The underlying operating system typically handles standard information types such as e-mail, calendar, and contacts, but it is left to individual applications to handle other types of data (e.g., updates from your favorite social networking site need to have an application running on the device). Unfortunately, these applications are often built in isolation and don't interact with each other or even integrate into the phone experience. In building Windows Phone, it was important for Microsoft to build an immersive user experience, rather than a set of disjointed applications. This needed to encompass all the features that make up Windows Phone, as well as the ability for third-party applications to be built to integrate into the same experience.

It was decided that Windows Phone should have a fresh start and not just the new Start that you'll see when you first unlock a Windows Phone. This should not just be about an ad hoc change to the way applications look, but a new language for communicating with users. What Microsoft came up with is what they refer to as the *Metro* design language, based on generations of refinement that have gone into the use of signposting, signaling, iconography, fonts, and layout in the transportation industry. The name *Metro* was chosen to reflect its heritage in the language used to efficiently guide people to their destinations.

If you examine signs used around and on buses, trains, and other forms of transportation, and at stations, airports, and other transportation hubs, you can clearly see a set of principles that govern them. Most of them have graphics that are instructional, yet minimal in design. Furthermore, the signs are typically universal and feature simple icons. The use of color is significant and plays an

important role alongside weight and style when applied to text. Unlike in a lot of software systems, where it is almost an afterthought, the use of different typography in signs can dramatically affect the readability and thus the effectiveness of those signs.

Using transportation signage as a base, the Windows Phone team collaborated with other teams within Microsoft in order to flesh out the design language. For example, the use of motion and animation in the Xbox, the use of navigation via content on the Zune and Media Center, and the intimacy of the ZuneHD interface were all sources of inspiration for Metro.

The resulting Metro design language is about being modern with a clean and fast user experience. It's about delivering an experience that is in motion and all about the content. Being authentically digital in design and using clear typography are also important throughout the Windows Phone experience.

Principles

In order to build a consistent experience across the Windows Phone platform, it was important not only to have a concept of a design language, but also to have a set of principles that would govern all decisions made in constructing the user experience. The following are the Metro design principles:

- Clean, light, open, fast
- World class motion
- Integrated hardware and software
- Content, not chrome
- Soulful and alive

Before you learn more about the Metro experience itself, take a moment to review these principles and more importantly think about how they can affect the way you build the user experience for your application. The first principle talks about the experience being "Clean, light, open, and fast." Although this doesn't limit itself to certain aspects of the interface, you can apply this principle by making sure your application isn't cluttered with icons and images and that any action the user takes is fast and responsive.

Just because screens on mobile phones have gotten larger over the last couple of years, this doesn't mean that you can pack more information onto a single screen. Doing so makes the content hard to read and difficult to navigate. Build a clean and light interface that displays only the important and relevant information to the user. This will necessitate larger and clearer fonts, which is where rich typography becomes important. The Metro design language celebrates typography and provides you with some rich fonts out-of-the-box for making your applications clear and easy to read.

The introduction of touch screens brought with it a need for haptic, or at least responsive, feedback. If you imagine touching the water in a glass or on a pond, you can immediately see the water ripple out from where you first touched the water. Instantaneous feedback, such as the changing of background color or a slight vibration of the device, gives the impression that the application is alive and awaiting the user's instructions. You will notice that motion is built into the core platform. The Start is made up of tiles that dynamically update and respond to being touched. Transitions provide

visual indicators to the user of not only where they are going, but where they have come from. The use of animation and other visual effects using motion is essential in building for Windows Phone. Within Windows Phone 7 the capacitive touch screen and other sensors enable gestures to control hardware accelerated animations and transitions which enhance the application. Hardware and software features blend together to provide a unified user experience.

Generations of applications across all manner of computer systems have adopted a windowing approach in which the border allows the user to control the window. On Windows Phone, there is no chrome! There is no border on the window, on controls, or even on content. Chrome takes up valuable screen real estate and for what, to illustrate that one piece of content has ended and the next has begun. If this is the case, use the content itself to indicate where those transitions are. For example, when you have adjacent images, instead of separating them by a few pixels, simply fade out the trailing edge of one image so that it's clear where the second image begins, as shown in Figure 1-3.

FIGURE 1-3

The first set of images in Figure 1-3 illustrates how images can literally be separated using empty space. Applying the content not chrome principle, this can be adapted to the second set of images, where the trailing edge, in this case the right edge, of the image is blurred or faded in order to make way for the next image.

At the end of the day, don't forget that you are building a software application for a mobile phone, which should be intimately attuned to its owner. Build applications that are soulful and alive, which offer a personalized and rewarding experience showing the information that matters most to the user with minimal excess. Bring the application to life by integrating the user experience with the rest of the device and making use of the platform's unique hardware capabilities to make it feel responsive to the user's gestures.

Applications should embody the three Red Threads of Windows Phone 7: Personal — your day, your way; Relevant — your people, your location; and Connected — your stuff, your peace of mind. We will investigate this theory further in Chapter 3.

User Experience

The Windows Phone user experience is the canonical representation of the Metro design language. However, having a design language isn't enough in itself to build a user experience. A user experience is governed by what you want to display and how users will interact with what is being displayed. For the Metro user experience, there were two guides that were used to direct the presentation of information: The user experience should focus on the individual and his or her tasks, and to help organize information and applications.

You have to ask the question, "Who was Windows Phone built for?" and, surprisingly, Microsoft has a remarkably well-defined answer. Windows Phone was designed for Life Maximizers and specifically Anna and Miles. *Life Maximizers* are defined as individuals who value the use of technology in their lives: They have both a busy personal life and professional life, yet they are more

settled than seeking and interact with their respective families. To empathize with such individuals and to understand their needs and wants, the personas of Anna and Miles were created. They are a pair of married, 38-year-old Life Maximizers who demand the most from their devices. As you investigate further, you learn that Anna is a part-time PR professional and a busy mother to whom balancing friends, work, and family is essential. Meanwhile, Miles is growing his architectural business and thus needs information available wherever he finds himself. Although these personas were created for the purpose of creating the Windows Phone experience, it is important for you to consider them when building your application. What would Anna or Miles want or expect?

START AND LOCK SCREENS

FIGURE 1-4

Now that you have heard about the Metro design language, it's time for you to see it in action as you navigate through the Windows Phone experience. It seems natural to begin with Start. However, you'll soon realize that the first thing you see on a Windows Phone device is actually the Lock screen. This is the informational screen that appears when the device is locked to ensure both the privacy of your data as well as to prevent you from accidentally dialing someone while your phone is in your pocket. Figure 1-4 illustrates the Lock screen showing the current date and time in a beautiful, clear white font overlaid on one of the default images that are available on the device. Both the time-out, whether there is a password for unlocking the device, and the background image can all be customized via Settings ➪ lock & wallpaper.

Once you unlock the Windows Phone, you enter the Start screen. As you can see from Figure 1-5, the Start screen is made up of a series of square tiles (there are some built in tiles such as the tile for the Pictures hub that take up two squares). These aren't just static blocks representing links to applications, in other words, icons similar to application icons on other mobile platforms. The tiles, also known as *live tiles*, are just that — they're alive and will not only dynamically update, but they'll also respond to your touch. The information displayed on Start should be what is relevant and important to you.

FIGURE 1-5

When you simply tap a tile, it appears to enlarge slightly before transitioning to the appropriate window. The right image shows what happens when you tap and hold one of the tiles. In this case, the tile enlarges even more, going into a mode in which you can drag the tile around the screen. As you move the tile, the other tiles will move themselves to allow you to drop the tile where you want it positioned. When you are done, simply tap in an area where there are no other tiles. Alternatively, if you no longer want a

tile to be pinned to the Start area, you can tap the pin icon. This will remove that tile from the Start area.

It's all very well to have these dynamic, adjustable tiles, but what happens to all those applications that aren't immediately visible on Start? Well, there are a couple of places where you can go to look for them. In the left image of Figure 1-6, you can see a Start that contains a large number of tiles and that has been scrolled up to reveal additional tiles. While not immediately obvious, Start can contain as many or as few tiles as you want. It's worth adjusting these tiles so that those that are important to you appear at the top of Start. Hitting the right arrow icon takes you to an alphabetic list of the available applications, shown in the right image of Figure 1-6.

FIGURE 1-6

Whereas the left side of Start is designed for glance-and-go information, the applications list has been designed to be easy to navigate in order to get you where you need to go as quickly as possible.

HUBS

As you step out of Start, you will most likely end up in one of six hubs: People, Pictures, Music + Video, Games, Office, and Marketplace. Each of these hubs is structured around a Panoramic view in which the screen acts as a small portal into a larger display. The panoramas are designed to be cyclic. The content loops around so that if you keep scrolling either to the right or the left you will see each of the content groups available in the panorama. In Figure 1-7, you can see the People hub, which shows your contacts in groups that represent Recent, All, and What's New.

FIGURE 1-7

The intent is that through the People hub you have access to information about your contacts that you care the most about. Updates to your contacts are drawn from social networking sites so that you won't miss what your friends and family are up to.

FIGURE 1-8

In Figure 1-8, you can see the Pictures hub, providing you with access to not only the various galleries you have for photos, but again a What's New area that draws on photo information from both your phone and your online photo stores. You'll notice with the panoramas that they are a great demonstration of content, not chrome. There is little in the way of navigation controls, yet through clever positioning of text and the content, it is intuitive for the user to scroll to the left and right to explore the hubs.

The Music + Video hub is very similar to the Pictures hub in structure. Building on the success of the ZuneHD, Microsoft has made every Windows Phone a Zune, and as such, the Music + Video hub is where you go to organize and play music and video on the device.

FIGURE 1-9

Not only did Microsoft integrate the Zune experience for media playback, but they've also made the Windows Phone a gaming device. Through the Games hub, you can access games on the device. Your Xbox live account is also accessible via this hub, as shown in the focus area of Figure 1-9. This figure illustrates how the device acts as a view into the panoramic experience. The header "ames" indicates that there is content to the left, while the cut off "C" on the right of the screen indicates that there is more to the right as well.

With all the features designed for consumers you may be wondering whether Windows Phone will make a good corporate device. The answer is that it will, indeed. The Office hub, shown in Figure 1-10, is where you can go to access your documents both from the device and via SharePoint.

FIGURE 1-10

The last hub, but definitely not the least, is Marketplace. As you can imagine, this is where users will go to purchase and download applications that you have written. The Marketplace will be split into genres such as music, applications, and podcasts. It will also support searching for applications via categories, as well as notifications for when there are application updates.

The Metro user experience introduces several new ways to present and navigate information. The hub concept is designed to group related information together and make it easy to glance at the most relevant information, such as your friends' recent social networking updates. Through the use of panoramas and an area much larger than the actual screen, hubs allow the personas Anna and Miles to quickly access frequently used information and updates, while still providing access to a significant volume of details and features that may be used less frequently. Although you can build your own Panoramic views, you need to be careful that you don't overuse them in cases where a single view would be more appropriate.

DEVELOPER LANDSCAPE

Building simple applications for Windows Mobile has always been a trivial exercise. You could simply open up Visual Studio, create a new project, and hit Debug or Run in order to see your application run on either an emulator or a connected device. Unfortunately, over time, support within Visual Studio and the associated .NET Compact Framework didn't keep pace with expectations in the market for rich applications that both looked good and were responsive to the user. In order to build sophisticated applications for Windows Mobile, you had to hunt down information on frameworks such as DirectDraw or OpenGL in the hope that they would allow you to build an application with a rich user experience. Even then there were non-trivial issues to resolve as different manufacturers decided not to provide driver-level support for them, or introduced device-specific functionality, or worse yet, bugs.

One of the goals with Windows Phone was to address these issues and provide a platform upon which developers could build both applications and games. The time to get up-and-running should be minimal, and yet the tools and frameworks need to be sophisticated enough to handle the most complex of user interfaces. Figure 1-11 provides a thousand-foot view of the development ecosystem for Windows Phone developers.

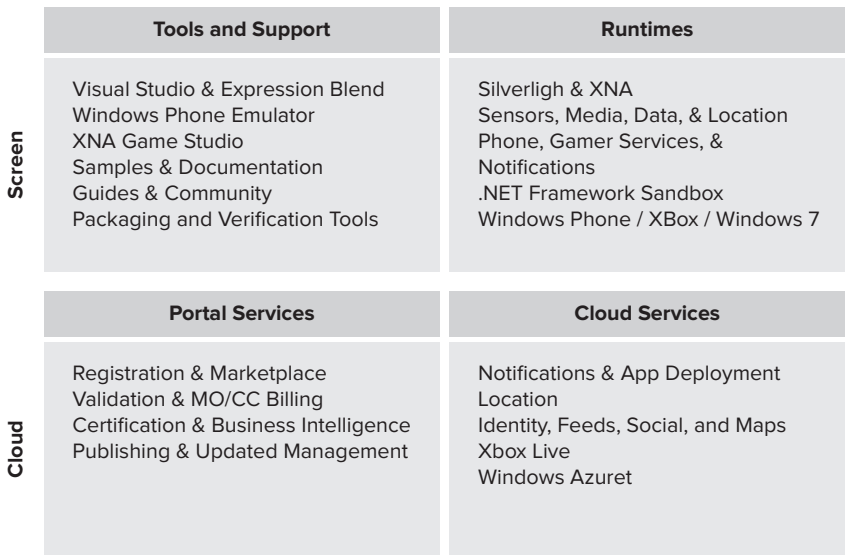

FIGURE 1-11

Let's start in the top-left corner with Tools and Support. For anyone who is already using Visual Studio 2010, you'll be aware that there is no support for doing Windows Mobile development. This was primarily a decision as to where to invest resources by Microsoft, and I think everyone will agree that it was a decision well made. The tooling for Windows Phone throughout both Visual Studio 2010 and Expression Blend dramatically reduces the time to build applications. In addition, the fact that these two products share the same solution and project structure means that both developers and designers can work in harmony on the same project at the same time. These tools will be covered in more detail in the next chapter, which shows how easy it is to get started with building your first application and debugging it on the Windows Phone Emulator.

Looking to the right, you'll see the Runtimes square. One of the most significant aspects of Windows Phone is that it fulfills the Microsoft three-screens goal. As a developer you can build an application or game and have it target one or more of the screens running Windows Phone, Xbox, or Windows 7. Unlike the .NET Compact Framework, which provided a reduced feature set compared to the desktop, the Silverlight and XNA runtimes available within Windows Phone are virtually identical to those you are already familiar with. Although you need to make a decision as to whether you're going to use Silverlight or XNA, you can still access a wide range of functionality from across the device no matter what technology you build your application or game in.

In building applications for Windows Phone, you are likely to want to access some of the cloud-based services that Microsoft has to offer. These include a notification service, for when application data changes and you want to notify the user; a location service, which integrates the device capabilities such as GPS with online services for resolving Wi-Fi locations; and access to the user's Xbox Live information from within your application. Going forward, expect further support for building Windows Phone applications that integrate with other cloud offerings such as Bing Maps and Windows Azure.

The last quadrant refers to the Developer Portal Services, which covers all the online services through which you as a developer interact in order to have your application certified and published via Marketplace. As Microsoft wants to ensure the highest quality of applications in Marketplace, there will be a more rigorous process for developing for Windows Phone. The online portal will be the point of reference for all your applications and will, hopefully, be the place where you go to receive monies generated by your application.

SUMMARY

Throughout this chapter, you have learned about the Windows Phone user experience, its origin in the transportation industry, and the birth of the Metro design language. In the coming chapters, you will see this language and its associated principles in action as you discover more about what it takes to build a Windows Phone application.